Table of Contents

kungeekUI

kungeekUI是一套集合了多个js组件的前端组件库。

在线阅读文档：在线阅读

使用指南

下载文件

文件源码地址：源码地址

引入文件

```
<!-- kungeekui.css  -->
<link rel="stylesheet" type="text/css" href="/skin/css/common/ku
ngeekui.css">

<!-- kungeekui.js  -->
<script src="/skin/script/kungeekui.js"></script>
```

*依赖关系

- `kungeekui` 基于 `Bootstrap` 与 `jQuery` ，使用前先确保页面引用了 `Bootstrap` 与 `jQuery`
- dateSelect基于 `momentjs` ，使用dateSelect之前请引入 `momentjs`
- treeSelect基于 `ztree` ，使用treeSelect之前请引入 `ztree`

Datagrid 数据表格

Datagrid 数据表格用于将后台数据通过异步加载到页面进行显示，并提供丰富的配置和方法，应对各种各样的表格的显示和操作需求。

使用说明

- 在页面中创建 `<table>` 标签，例子如下:

```
<table id="customTable" class="table kungeekui-datagrid"
    kg-data-source="qzkhxxPageData.do"
    kg-data-option="hasOrder:false;hasCheckBox:true;paging:true;
fieldDrag:true;defaultPage:1">
        <thead>
            <tr>
                <th kg-data-option="field:qzkhMc;width:250px">潜
在客户名称</th>
                <th kg-data-field="lxr" kg-data-width="65">联系人
</th>
                <th kg-data-field="mphone">联系人电话</th>
                <th kg-data-field="sjlx">保护状态</th>
                <th kg-data-field="yssjLy">数据来源</th>
            </tr>
        </thead>
</table>
```

- 相关说明：
- 指定table元素的 `class` 包含 `kungeekui-datagrid` ，table元素将自动初始化；
- 通过标签属性为表格设置初始配置；
- 初始化后要操作整个表格请选择table的父级 `.tableContainer` 元素；
- 分页数据异步接口需要返回PageResult分页信息，PageResult结构：

```
{
"pagedResult": {
 "data": [//数据集合
   {
     "fieldName":"fieldVal"
   }
 ],
 "total": 0,//总数据条数
 "info": {
   "next": 1,//下一页是第几页
   "prev": 1,//上一页是第几页
   "totalPage": 1,//总页数
   "index": 0,
   "pageIndex": 1,//当前第几页
   "pageSize": 10,//每页显示多少行
   "last": 0,//最后一页是第几页
   "first": 1,//第一页是第几页
 }
}
}
```

属性

设置

通过设置 `kg-data-option` 或者 `kg-data-[param]` 为表格或列设置属性。

```
kg-data-option="hasOrder:false;hasCheckBox:true;paging:true;fieldDrag:true;defaultPage:1"

or

kg-data-width="25%"
```

- `kg-data-[param]` 可以写在 `kg-data-option` 里面，因此可以不用设置 `kg-data-[param]`
- 一旦设置了 `kg-data-[param]` ，那么 `kg-data-[param]` 的值将会取代 `kg-data-option` 中对应的属性值

属性表

table 表格属性：

属性名	类型	默认值	说明
width	String	'100%'	表格宽度
height	String	'auto'	表格高度
minTrNums	Number	10	表格最小行数
title	String	null	表格标题（未完善）
source	String	null	表格数据源地址
autoLoadSoure	Boolean	true	是否自动第一次加载数据源
defaultPage	Number	1	初始化时加载第几页
paging	Boolean	false	是否要分页
hasOrder	Boolean	true	是否需要排序列
hasCheckBox	Boolean	false	是否需要行Checkbox
onlyCheckOne	Boolean	false	是否单选
fixedThead	Boolean	true	是否固定表头
extraClass	String	'table'	表格额外的class属性
trAnimation	String	null	tr渲染动画名（animate.css）
editField	Boolean	true	是否可以显示/隐藏列
fixedOrderField	Boolean	true	是否左边固定排序列
fixedCheckBoxField	Boolean	true	是否左边固定chengbox列
fieldDrag	Boolean	true	是否允许列宽度拖动
fieldOrder	Boolean	true	是否需要列排序
minFieldWidth	Number	70	最小列宽
codeUrl	String	某url	码表地址

th 列属性：

属性名	类型	默认值	备注
width	String	'initial'	列宽度
field	String	''	列属性名
align	String	'center'	列对齐方式
newline	Boolean	false	内容是否换行
canSelect	Boolean	true	文本是否可以被选择
fixed	Boolean/String	false	是否固定列
			true/'left'为左边固定，'right'为右边固定
orderName	String	''	排序的名字
			如果有值代表这个列可以排序，并且每次请求会将此名传到后台
rule	String	null	本列输出的值采用的变换规则
			`变换前的值` -> `变换前的值[rule]()`
code	String	''	码表参数
			用于码表接口提供参数，返回值映射到最后的输出值

方法

使用

通过jquery选择器选择Dategrid 实例标签进行方法调用:

```
$('#table').getSelected();
```

方法表

方法名	说明	使用方法
getSelected	获取当前选择项目	查看
cancelSelected	取消所有选择	查看
addField	添加列	查看
modifyField	修改列显示	查看
showField	显示列	查看
hideField	隐藏列	查看
addBottomTr	添加表格底部行	查看
gotoPage	跳转到第几页	查看
getPage	获取当前是第几页	
getPageSize	获取当前每页显示多少条	
getAllData	获取所有数据	
getSource	获取当前数据源	
getTotalData	获取总共多少条数据	
reloadUrl	重新加载数据源	查看
putData	把数据输入表格	查看
loadData	异步加载数据把数据输入表格	查看
refresh	刷新表格	
updateTableWidth	更新表格宽度	
mask	显示或隐藏表格遮罩	查看

---

getSelected

获取当前选择项目。

```
var a = $('#table').getSelected();
```

cancelSelected

取消所有选择。

```
$('#table').cancelSelected();
```

addField

添加列。

```
$('#table').addField('操作', function(trData) {
    if (trData.param == "编辑") {
        return '<button type="button">"编辑按钮"</button>'
    }
    return '<button type="button">"不是编辑按钮"</button>'
});
//第一个参数代表添加的列对应的字段名
//第二个函数参数返回内容将显示到表格中
//第二个函数中trData代表每条数据


$('#table').addField({title:'操作',width:200}, function(trData) {
    if (trData.param == "编辑") {
        return '<button type="button">"编辑按钮"</button>'
    }
    return '<button type="button">"不是编辑按钮"</button>'
});
//可以指定添加列的宽度


$('#table').addField({title:'操作',width:200}, function(trData) {
    if (trData.fieldName == "编辑") {
        return '<button type="button">"编辑按钮"</button>'
    }
    return '<button type="button">"不是编辑按钮"</button>'
},3);
//最后一个参数将代表添加的列的位置在表格的第三列
```

modifyField

操作列显示值。

```
$('#table').modifyField('fieldName', function(trData) {
  return trData.fieldName + '修改'
});
//第一个参数代表修改的列对应的字段名
//第二个函数参数返回内容将显示到表格中

$('#table').modifyField([{
    field:'fieldName1',
    ruleFn:function(trData){
        return trData.fieldName1 + '修改'
    }
},{
    field:'fieldName2',
    ruleFn:function(trData){
        return trData.fieldName2 + '修改'
    }
}])
//一次方法修改多个列
```

showField

显示列。

```
$('#table').showField('fieldName');
```

hideField

隐藏列。

```
$('#table').hideField('fieldName');
```

addBottomTr

添加表格底部行。

```
$('#table').addBottomTr(function (tableData) {
    return '<td></td>';
})
```

gotoPage

跳转到第几页。

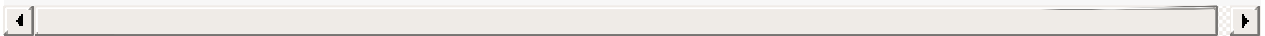
```
$('#table').gotoPage(2);
```

reloadUrl

重新加载数据源，即重新加载表格的source。

```
$('#table').reloadUrl('callAndSmsCenter.do?callAndSmsData&gsId=2'
);

$('#table').reloadUrl('callAndSmsCenter.do?callAndSmsData&gsId=2'
,true);
//第二个参数为true将返回当前页的数据，否则是第一页的数据
```



putData

将数据输入表格显示，前提是表格没有source配置。

```
var data = [{
    id:1,
    fieldName:fieldVal
},{
    id:2,
    fieldName:fieldVal
}]
$('#table').putData(data);
```

loadData

异步获取数据后，将数据输入表格显示，相当异步获取数据后执行putData。前提是表格没有source配置。

```
$('#table').loadData('callAndSmsCenter.do?callAndSmsData&gsId=2'
);
```

mask

显示表格遮罩，和loading不同的是没有loading图标。

```
$('#table').mask('show');//显示表格遮罩

$('#table').mask('hide');//隐藏表格遮罩
```

事件

使用

通过jquery选择器选择Dategrid 实例标签进行事件调用:

```
$('#table').on('rendered',function(e,data){
    //do something
});
```

事件表

事件名	执行时间	例子
rendered	表格渲染完成后	查看
loaded	表格异步请求数据完成后	查看

rendered

```
$('#table1').on('rendered',function(e,data){
    console.log('表格渲染完成',data)
});
```

loaded

```
$('#table1').on('loaded',function(e,data){
    console.log('表格请求数据完成',data)
});
```

ShowDialog 模态框插件

ShowDialog 模态框插件创建页面模态框，可以通过选择器选择页面元素创建本页模态框，或者通过指定地址创建iframe模态框。

ShowDialog 模态框插件提供丰富的配置和方法。

使用说明

- 通过new ShowDialog创建showDialog对象：

```
var showDialog = new ShowDialog({
        title: '标题',
        content: '#modal'
        })
```

- 相关说明：
- 一个页面只能同时显示一个ShowDialog模态框，也就是新的模态框显示后，其他所有模态框将自动隐藏；
- 为了节省资源，多个showDialog对象公用一个公共的modal元素，所有showDialog对象的content元素都将存在一个全局的modal中。

属性

设置

创建showDialog对象时传入相应配置。

```
showDialog = new ShowDialog({
                title: '标题',
                width: 500,
                content: '#resetPwd'
                //...
            });
```

属性表

table 表格属性：

属性名	类型	默认值	说明
width	String,Number	'90%'	模态框宽度
height	String,Number	"	模态框高度
title	String	"	模态框标题
content	String	"	模态框内容 详情
isIframe	Boolean	false	是否Iframe模态框 详情
slideInRight	Boolean	false	是否从右边滑出
buttons	Array	[{'取消按钮'}]	模态框底部按钮 详情
extraStyle	Object	{}	额外css样式
hasDefaultBtn	Boolean	true	是否有默认取消按钮
animated	String	'fadeInDown'	模态框弹出动画
animatedOut	String	'fadeOutUp'	模态框关闭动画

content 属性介绍：

- 选择器写法：在第一次show事件执行时，会把$('#el')元素移动到模态框里。

注意元素移动时会有短暂的时间会选择不到，第一次时应该是对元素的操作完成之后再调用show。

```
showDialog = new ShowDialog({
                content: '#el'
            });
```

- 字符串写法：会在模态框里创建文本元素。

  ```
  showDialog = new ShowDialog({
                  content: '一些字符'
              });
  ```

- url写法：如果isIframe为true，那么content属性则为iframe的url，也可以在reloadUrl方法里指定url。

  ```
  showDialog = new ShowDialog({
                  content: 'url'
              });
  ```

isIframe 属性介绍：

```
showDialog = new ShowDialog({
                isIframe: true
            });
```

指定isIframe为true，会创建一个iframe元素到模态框里:

```
<iframe id="{showDialogId}" name="modalFrame" src="" frameborder=
"0" width="100%" height="100%"></iframe>
```

已自动设置iframe的样式、加载、尺寸、加载动画。

buttons 属性介绍：

```
showDialog = new ShowDialog({
                buttons: [{
                  text: '保存',
                  event: function() {
                      //...
                  },
                  btnStyle: 'btn-success'}]
            });
```

buttons是指定模态框底部的按钮组。 如果hasDefaultBtn为false，则没有默认的取消按钮：

```
var button = {
    text: '取消',
    event: function () {
        kungeekui.$modalDom.modal('hide');
    },
    btnStyle: 'btn-default'
}
```

如果hasDefaultBtn为true,则会在指定的buttons上堆加取消按钮。

button对象属性：

属性名	类型	默认值	备注
text	String	''	按钮文字
event	Function	null	按钮事件
btnStyle	String	'btn-default'	按钮样式

方法

使用

showDialog对象调用方法：

```
var showDialog = new ShowDialog({
        title: '标题',
        content: '#modal'
        })
showDialog.show();
```

方法表

方法名	参数	返回类型	备注	使用方法
show	null	null	弹出模态框	查看
hide	null	null	隐藏模态框	查看
addArgument	(key,val)	null	为模态框设置变量	查看
getArgument	(key)	(val)	获取模态框变量值	查看
setTitle	(title)	null	设置模态框标题	查看
reloadUrl	(url)	null	重载Iframe模态框	查看

show

弹出模态框。

```
showDialog.show();
```

hide

隐藏模态框。

```
showDialog.hide();
```

addArgument

为模态框设置变量，相同变量值会覆盖，可设置多个变量值。

```
showDialog.addArgument('id',2);
```

getArgument

获取模态框变量值。

```
showDialog.getArgument('id');//2
```

setTitle

设置模态框标题。

```
showDialog.setTitle('客户信息');
```

reloadUrl

```
showDialog.reloadUrl('url');
```

事件

使用

showDialog对象调用on方法：

```
showDialog.on('show',function(){
    //执行一些方法
});
```

事件表

事件名	执行时间
show	模态框显示前
shown	模态框显示后
hide	模态框隐藏前
hidden	模态框隐藏后

DateSelect 日期选择插件

DateSelect 是一个功能丰富的日期框选择插件，支持单独或同时选择年度、季度、月度、日、时、分，并且有日期格式设置，限制日期期限等功能。

使用说明

- dateSelect基于 `momentjs`
- 在页面中创建文本 `<input>` 标签并指定class包含 `kg-dateSelect` ，input标签将自动渲染为dateSelect插件，例子如下:

```
<input type="text" name="date" class=" kg-dateSelect"
        kg-select-type="D"
        kg-select-format="YYYY-MM-DD"
        kg-select-formatsimple="false">
```

- 或者手动初始化:

    ```
    <input type="text" name="date" id="kg-dateSelect">
    ```

    ```
    $('#kg-dateSelect').dateSelect({option});
    ```

属性

设置

通过设置 `kg-data-[param]` 为表格或列设置属性:

```
kg-select-type="D"
```

或者通过初始化传入配置为表格设置属性:

```
$('#kg-dateSelect').dateSelect({selectType:'D'});
```

属性表

dateSelect 表格属性：

属性名	类型	默认值	说明
selectType	String	'YSMD'	日期选择类型
format	String	'YYYY-MM-DD'	日期显示样式
gap	String	' 至 '	日期分隔符
formatSimple	Boolean	true	是否显示为中文日期
readonly	Boolean	true	是否允许直接修改input(未开放)
clearBtn	Boolean	true	是否有清空按钮
minDate	String	''	最小日期
maxDate	String	''	最大日期

方法

使用

通过jquery选择器选择DateSelect实例标签进行方法调用:

```
$('#dateInput').getDate();
```

方法表

方法名	参数	返回	备注	使用方法
getDate	null	Object	获取选择日期信息	查看
clearDate	null	jQ元素	清空日期	
setMinDate	String	jQ元素	设置最小日期	查看
setMaxDate	String	jQ元素	设置最大日期	查看

getDate

获取选择日期信息。

```
$('#dateInput').getDate();
//Object：{
//            val:'2018年' ,//input的value
//            startDate: '2018-01-01',//具体到天的开始日期
//            endDate:'2018-12-31',//具体到天的结束日期
//            type:'Y'//选择类型
//         }
```

setMinDate

设置最小日期。

```
$('#dateInput').setMaxDate('2018-01-01');
```

setMaxDate

设置最大日期。

```
$('#dateInput').setMaxDate('2018-12-01');
```

事件

使用

通过jquery选择器选择DateSelect实例标签进行事件调用:

```
$('#dateInput').on('change',function(e,data){
    //do something
});
```

事件表

事件名	执行时间	例子
change	改变日期选择后	查看
blur	失去焦点后	查看

change

```
$('#dateInput').on('change',function(e,dateVal){
    console.log('改变日期选择后',dateVal)
});
```

blur

```
$('#dateInput').on('blur',function(e,dateVal){
    console.log('失去焦点后',dateVal)
});
```

SelectTree 日期选择插件

SelectTree 是一个功能丰富的下拉树插件，支持多种功能，基于 `ztree` 。

使用说明

- selectTree基于 `ztree`
- 在页面中创建文本 `<input>` 标签:

  ```
  <input type="text" name="date" id="kg-selectTree">
  ```

  然后通过方法驱动：

  ```
  $('#kg-selectTree').selectTree({optionObject});
  ```

属性

或者通过初始化传入配置为表格设置属性:

```
$('#kg-selectTree').selectTree({zNodes:[]});
```

属性表

selectTree 表格属性：

属性名	类型	默认值	说明
zNodes	Array	[]	节点数据
setting	Object	{}	ztree的setting
defaultSelectedNodes	Array	[]	默认选择的节点数据
showTopDrop	Boolean	false	是否显示常用选项
defaultTopNodes	Array	[]	常用选项节点数据
topDropLabel	String	'全部选项'	显示文本

方法

使用

通过jquery选择器选择DateSelect实例标签进行方法调用:

```
$('#selectTree').getTreeObj();
```

方法表

方法名	参数	返回	备注	使用方法
getTreeObj	Null	Object	获取ztree对象	
setSelectTreeNodes	Array	jQ对象	设置Nodes数据	
setSelectedTreeNodes	Array	jQ对象	设置选择的Nodes数据	
getSelectTreeNodes	Null	Array	获取选择的Nodes数据	
getSelectTreeData	Null	Array	获取选择的数据	
clearSelectTree	Null	jQ元素	清空设置的Nodes	
selectTreeNodeByDataParam	(val, param, nodes)	jQ元素	通过Node节点data对象中属性设置默认值	-

事件

使用

通过jquery选择器选择selectTree实例标签进行事件调用:

```
$('#selectTree').on('change',function(){
    //do something
});
```

事件表

事件名	执行时间	例子
change	改变选择后(单选模式)	查看
check	改变选择后(多选模式)	查看

change

```
$('#selectTree').on('change',function(e,dateVal,ztreeId){
    console.log('改变选择后',dateVal,ztreeId)
});
```

check

```
$('#selectTree').on('check',function(e,dateVal,ztreeId){
    console.log('改变选择后',dateVal，ztreeId)
});
```

FormValid 表单验证插件

FormValid 是一个精巧的课配置表单验证插件。

使用说明

- 为页面中的form标签指定属性:

  ```
  <form id="qzkhForm" class="form-horizontal crm-form" kg-form
  -valid="true">
  ```

  或者

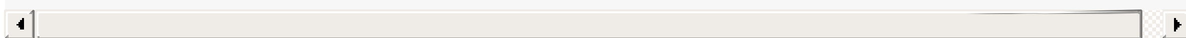
  ```
  new kgValidForm(formElement);
  ```

  或者

  ```
  $('#form').kgValidForm();
  ```

- 然后通过标签属性为表单元素指定验证条件：

  ```
  <input type="text" class="form-control input-sm" id="eMail"
  name="eMail" placeholder="请输入电子邮箱" kg-data-valid="email">
  ```

  添加验证条件后会根据表单元素的事件响应实时监控表单元素值的合法性，给与用于提示。

- 调用form标签元素方法验证:

  ```
  $('#qzkhForm').kgValidParam();//验证通过返回true；失败返回false
  ```

属性

验证条件

- 通过标签属性为表单元素指定验证条件：

```
<input type="text" kg-data-valid="email">
```

验证条件表：

验证条件说明	验证条件
不能为空	notNull
纯数字	numeric
正整数	positive
整数	integer
浮点数	decimal
正浮点数	isdecimal
Email地址验证	email
部门职员，用户信息 - Email地址验证（保持与sap一致）	email2
纯字母	alpha
字母和数字	alphaNumeric
字母数字字符、下划线和破折号	alphaDash
必须是正整数	natural
必须是非零的整数	naturalNoZero
必须包含一个有效的IP	ip
必须是数字，-，空白字符、空格、制表符和换行符	numericDash
必须包含一个有效的网址	url
必须包含日期格式YYYY-MM-DD	date
必须是中文字符	chineseCharacter

必须是有效手机号码或者固话号码	mobileOrTelephone
必须是有效手机号码	mobilePhone
必须是有效固话号码	telephone
必须是有效QQ号码	qq
必须是有效的邮政编码	postcode
必须是有效身份证号码	idCard
不能包含特殊字符	specialCharacter
必须是有效微信号	weixin
必须是中文字符或者英文字符	chineseAlpha
必须是中英字符或者中文符号	chineseSign
必须是正数或两位正浮点数	money
必须是非零正数或两位正浮点数	moneyNoZero
必须是http或者https开头的网址	webUrl
crm - 客户名称匹配规则	crmCustomName
crm - 企业联系人匹配规则	crmCompanyContactsName

文本提示表：

验证条件	文本提示
notNull	输入值不能为空！
specialCharacter	特殊符号只能包含小括号！
mobileOrTelephone	请输入正确的电话号码！
mobilePhone	请输入正确的手机号码！
email	请输入正确的邮箱地址！
telephone	请输入正确的固话号码！
chineseAlpha	请输入中文字符或者英文字符！
alphaNumeric	请输入英文字母或数字！
url	请输入一个有效的网址！
chineseCharacter	必须是中文字符！
positive	请输入大于0的正整数！
numeric	请输入正整数！
crmCustomName	只能包含中文/英文、中英文小括号、阿拉伯数字！
crmCompanyContactsName	只能包含中文/英文、小圆点！

方法

使用

通过jquery选择器选择FormValid实例标签进行方法调用:

```
$('#form').addNodeValid();
```

方法表

方法名	参数	返回	备注
addNodeValid	(target, defaultValue)	jQ对象	动态新增验证节点
removeNodeValid	(target)	jQ对象	清除节点验证绑定（即该节点不再有验证行为）
setDefultValidValue	(defaultValue)	jQ对象	设置不进行接口验证的值
clearDefultValidValue	\	jQ对象	清除不进行接口验证的值
clearValidHints	(nodes)	jQ对象	重置验证插件，清空错误验证
kgValidParam	\	jQ对象	验证form表单所有输入框

CRM平台公用方法

CRM平台前端内置多个公用方法，便于高速开发，这些方法一般放在crm.portal.js里。

公共方法

initCodeSelectPicker()

根据码表更新下拉框的数据 基于selectPicker。 使用方法：

```
<select id="htlx" name="htlx" class="form-control selectpicker"
kg-data-code="crm_htlx">
     <option value="">--请选择--</option>
</select>
```

源码：

```
if ($('table.kungeekui-datagrid').length < 1 && ('select.selectp
icker').length > 0) {//自动执行
    initCodeSelectPicker();
}
function initCodeSelectPicker() {
    $('select.selectpicker').each(function () {
        var code = $(this).attr('kg-data-code');
        if (!code) {
            return;
        }
        if (top.codeMapperData[code]) {
            initSelect($('.selectpicker[kg-data-code="' + code +
'"]'), top.codeMapperData[code]);
            return
        }

        !function (code) {
            $.ajax({
```

```
                type: 'get',
                url: "/portal/crm/infra/ftspcode.do?query&code="
 + code,
                success: function (res) {
                    if (res && res[0]) {
                        top.codeMapperData[code] = res[0];
                        initSelect($('.selectpicker[kg-data-code
="' + code + '"]'), top.codeMapperData[code]);
                    }
                }
            });
        }(code)

        function initSelect(el, data) {
            if (el.get(0).hasInitedSelect) {
                return
            } else {
                el.get(0).hasInitedSelect = true;
            }
            for (i in data) {
                var $option = $('<option value="' + i + '">' + d
ata[i] + '</option>');
                el.append($option);
            }
            el.selectpicker('refresh');
        }
    })
}
```

redirectTab(name,url)

首页一级菜单的跳转。使用方法：

```
redirectTab('加盟商合同审核');
redirectTab('加盟商合同审核', '/portal/htContract.do?htSpPage');//
url优先级大于name
```

源码：

```
window.redirectTab = function (name, url) {
    if (!name) {
        return;
    }
    if (!window.top.$) {
        return
    }
    var $$ = window.top.$;
    $$('.submenu-label').each(function () {
        if ($(this).text() == name) {
            var li = $(this).parent('a').parent('li');
            $$('.openable').removeClass('open');
            $$('.openable .submenu').hide();
            $$('.openable .submenu').find('li').removeClass('act
ive');
            if (!$$('.sidebar-menu').hasClass('sidebar-mini')) {
                li.parent('ul').show();
            }
            li.parent('ul').parent('li').addClass('open');
            li.addClass('active');
            if (!url) {
                li.trigger('click');
            }
            return
        }
    })
    if (url) {
        window.location.href = url
    }
}
```

refreshPage(onlyTable)

刷新页面。

```
refreshPage();//刷新整个页面
refreshPage(true);//只刷新页面中的数据表格
```

tableQuery(form, table)

数据表格对应表单的查询。

```
tableQuery('#form', '#table');//根据Id为form的表单条件查询Id为table
的数据表格
```

源码：

```
window.tableQuery = function (form, table) {
 if (!form || !table) {
     return false
 }
 if (!$(form).length || !$(table).length) {
     return false
 }
 var url = $(table).getSource() ? $(table).getSource().setParams
ByUrl($(form).serializeArray().map(function (obj) {
     var $node = $(form).find('*[name=' + obj.name + ']');
     if ($node.hasClass('kg-selectTree')) {
         var _treeData = $node.getSelectTreeData()[0];
         obj.value = _treeData ? _treeData.data[$node.attr('data
-tree-query')] : '';
     }
     return obj;
 })) : '';
 $(table).reloadUrl(url);
}
```

initIndustryDrop()

初始化行业数据（下拉树）（用赋值window._default_hyDm 处理JSP方式的默认数据）

```
function initIndustryDrop(url, params) {
    var $industry = $('input[data-type=industry]');
    if (!$industry.length)
```

```
        return;
    $.getJSON(url || '/skin/cache/crm_hyfl.json', params || {},
function (result) {
        window._industryData = [{
            code: '',
            name: '请选择',
            id: '',
            pId: '1'
        }].concat(result.data || []);
        $industry.each(function () {
            var $this = $(this);
            var _zNodes = _industryData.map(function (obj) {
                return {
                    id: obj.id,
                    pId: obj.pId,
                    name: obj.name,
                    open: false,
                    data: obj
                };
            });
            var _defaultTopNodes = new Array(11);
            _zNodes.forEach(function (obj) {
                var _index = ['请选择', '批发业', '零售业', '租赁业'
, '商务服务业', '建筑安装业', '软件和信息技术服务业', '制造业', '专业技术
服务业', '农、林、牧、渔业', '居民服务、修理和其他服务业'].indexOf(obj.n
ame);
                _index >= 0 && _defaultTopNodes.splice(_index, 1
, obj);
            });
            $this.addClass('kg-selectTree').selectTree({
                zNodes: _zNodes,
                setting: {
                    callback: {
                        beforeClick: function (treeId, treeNode,
 clickFlag) {
                            $('.kg-selectTree-dropdown').hide();
                            return true;
                        }
                    }
                },
```

```
                defaultSelectedNodes: [_zNodes[0]],
                showTopDrop: true,
                defaultTopNodes: _defaultTopNodes
            });
            if (window._default_hyDm) {
                $industry.selectTreeNodeByDataParam(window._defa
ult_hyDm, 'code');
            }
        });
    });
}
```

reinitIframe()

重置Iframe高度

```
window.reinitIframe = function () {
    var $iframe = $('iframe.crm-iframe');
    $iframe.each(function () {
        this.height = "100%";
        if (!this.$iframeWrap) {
            this.$iframeWrap = $('<div class="crm-iframe-wrap"><
/div>');
            $(this).wrap(this.$iframeWrap);

        }
        if (!$(this).is(':visible')) {
            return
        }
        var height = $(this).contents().height();
        $('.crm-iframe-wrap').height(height);
    })
}
```

原生扩展

```
// 字符串扩展，属性字符串转对象 规则 : 'width':'200px'  -> {width:'20
```

```
0px'}
String.prototype.toObject = function () {
    return JSON.parse('{' + this.replace(/\s*['"]*([^'":;\s]*)['
"]*\s*:+\s*['"]*([^'":;\s]*)['"]*\s*;*/g, '"$1":"$2",').replace(
/,{1}$/g, '') + '}');
};

// 字符串 toBool 说明："false" to false
String.prototype.toBool = function () {
    return (/^true$/i).test(this);
};

//判断是否email
String.prototype.isEmail = function () {
    return /^\w+((-\w+)|(\.\w+))*\@[A-Za-z0-9]+((\.|-)[A-Za-z0-9
]+)*\.[A-Za-z0-9]+$/.test(this);
};

//检查电话号码
String.prototype.isPhoneCall = function () {
    return /(^1[3|4|5|7|8]\d{9}$)|(^0\d{2,3}-?\d{7,8}$)/.test(th
is);
};

//HTML字符串转义
String.prototype.toHtmlCode = function () {
    var _temp = document.createElement('div');
    _temp.innerHTML = this.replace(/((?:&))\s+((?:#))/g, '$1$2')
;
    return _temp.innerText;
};

//转换百分比
String.prototype.toLevel = function () {
    var temp = /[0-9]+(\.[0-9]+){0,1}/.exec(this);
    if (!temp)
        return '';
    var _str = Number(temp[0]).toString();
    var _index = _str.indexOf('.');
    var _chart = [];
```

```
    for (var i = 0; i < _str.length; i++) {
        if (_str[i] == '.')
            continue;
        _chart.push(_str[i]);
    }
    _index += 2;
    if (_index < 2 || (_index - _chart.length > 1))
        return _chart.concat(['0', '0']).join('').replace(/^[0]+
/g, '');
    if (_chart.length < _index)
        return _chart.concat(['0']).join('').replace(/^[0]+/g, ''
);
    if (_chart.length > _index)
        _chart.splice(_index, 0, '.');
    return _chart.join('').replace(/^[0]+/g, '');
};

//获取链接字符串中参数
String.prototype.getParamByUrl = function (name) {
    var _name = new RegExp(name + '=([^&]{0,})').exec(decodeURI(
this));
    return (_name && _name.length) ? _name[1] : null;
};

//生成链接参数对象
String.prototype.setParamByUrl = function (name, value) {
    var _paramStr = /[^]*\?([^?]+)$/.exec(decodeURI(this));
    var _this = this;
    if (!_paramStr && _this.indexOf('?') < 0) {
        _this += '?';
    }
    var _param = name + '=' + value;
    if (this.getParamByUrl(name) == null) {
        return _this.concat(((_paramStr && _paramStr.length) ? '
&' : '') + _param);
    }
    return _this.replace(new RegExp(name + '=([^&]{0,})'), _para
m);
};
```

```
String.prototype.setParamsByUrl = function (array) {
    if (!array || !array.push)
        return this;
    var _this = this;
    array.forEach(function (obj) {
        _this = _this.setParamByUrl(obj.name, obj.value);
    });
    return _this;
};

//将数字格式的字符串转换为时间格式的字符串
String.prototype.toDateFormat = function (format) {
    return /^\d{4,8}$/.test(this) ? this.replace(/\B(?=\d{2}$)/,
 format || '-').replace(/(^\d{4})\B(?=\d)/, '$1'.concat(format |
| '-')) : this.toString();
};

//转换数字字符串位数，默认2位，小于则补0，大于则截取
String.prototype.toFixDigit = function (number, last) {
    number = typeof number === 'number' && number > 0 ? number :
2;
    var _this = this.toString();
    if (!/^[\d]+$/.test(_this) || new RegExp('^[\\d][' + number
+ ']$').test(_this))
        return _this.toString();
    if (_this.length > number)
        return !last ? _this.substring(_this.length - number, _t
his.length) : _this.substring(0, number);
    while (number - _this.length > 0) {
        _this = !last ? '0'.concat(_this) : _this.concat('0');
    }
    return _this;
};

//金额字符串处理
String.prototype.toMoney = function () {
    var _regx = /[-]{0,1}[0-9]+(\.[0-9]+){0,1}/.exec(this);
    if (!_regx)
        return '0.00';
    var _temp = _regx[0].split('.');
```

```
    _temp[0] = _temp[0].replace(/\B(?=(\d{3})+(?!\d))/g, ',');
    _temp[1] = _temp[1] || '00';
    _temp[1].length < 2 && (_temp[1] += '0');
    return _temp.join('.');
};
Number.prototype.toMoney = function () {
    return this.toString().toMoney()
}
```

jQuery 扩展

crm_clearForm

清空表单数据

```
$('#form').crm_clearForm();
```

setLoader

显示/隐藏原生加载遮罩

```
$('#div').setLoader(true);//显示加载遮罩
$('#div').setLoader(false);//隐藏加载遮罩
```

源码：

```
$.extend(true, $.fn, {
    setLoader: function (status) {//显示加载蒙层
        return this.each(function () {
            try {
                if ($(this).css('position') != 'absolute' || $(this).css('position') != 'relative' || $(this).css('position') != 'fixed') {
                    $(this).css('position', 'relative');
                }
```

```
            } catch (e) {
            }
            if (!this.$loader) {
                this.$loader = $('<div class="kg-loader"><div class="kg-loader-icon"><i class="fa fa-spinner fa-spin fa-3x fa-fw" style="color:#0dbbe5"></i></div></div>');
                if ($(this).is('body')) {
                    this.$loader.css('position', 'fixed');
                }
                $(this).append(this.$loader);
            }
            if (status) {
                this.$loader.fadeIn();
            } else {
                this.$loader.fadeOut();
            }
        })
    },
    crm_clearForm: function () {//清空元素内表单元素值
        return this.each(function () {
            $(this).clearValidHints();
            $(this).find('input').not('input[disabled],input[readonly]').val('');
            $(this).find('input[type="checkbox"]').not('input[disabled],input[readonly]').prop('checked', false);
            $(this).find('input[type="radio"]').not('input[disabled],input[readonly]').prop('checked', false);
            $(this).find('input.kg-dateSelect').val('');
            $(this).find('textarea').not('textarea[disabled],textarea[readonly]').val('');
            $(this).find('select').not('select[disabled],select[readonly]').val('');
            $(this).find('select.selectpicker').selectpicker('refresh');
        })
    }
})
```

CRM平台默认方法

CRM平台前端内置多个默认处理程序，便于高速开发，这些方法一般放在crm.portal.js里。

统一异步请求前处理

在这里统一处理网站的异步请求。

```
window._token = ($("meta[name='_csrf']").length ? $("meta[name='
_csrf']") : $(top.document).find("meta[name='_csrf']")).attr("co
ntent");
    window._header = ($("meta[name='_csrf_header']").length ? $(
"meta[name='_csrf_header']") : $(top.document).find("meta[name='
_csrf_header']")).attr("content");
    headers[_header] = _token;
    $(document).ajaxSend(function (e, xhr) {
        if (xhr.setRequestHeader) {//设置请求头
            xhr.setRequestHeader(_header, _token);
            xhr.setRequestHeader('sendType', 'ajax');
        }
        try {
            xhr.done(function (data) {//超时处理
                try {
                    data = typeof data == 'string' ? JSON.parse(
data) : data;
                } catch (e) {
                }
                if (data.status == 'timeout') {
                    top.location.href = top.location.href;
                    return;
                }
            });
        } catch (e) {
            console.log('xhr不存在或者xhr没有done方法')
        }
    });
```

表单

表单由多个程序处理，负责不同的功能

```
//这里处理表单只有一个name时button触发的表单默认事件
$('form.crm-form').on('submit', function () {
    return false
});
```

```
//绑定查询表单的回车查询
$('form').on('keyup', function (e) {
    if (e.keyCode == 13) {
        var $query = $(this).find('*[data-type="query"]');
        $query.length ? $query.click() : (event.currentTarget.id
 == 'myqueryForm' ? myquery() : (window.query || function () {
        })());
    }
});

//限制电话类型输入框长度和内容
$(document).on('input', "input[name=dhQuery],input[name=mphone],
input[name=phoneNum],input[name=dhhm],input[name=osPhone],input[
name=phone],input[data-type=phone]", function () {
    //如果是固定电话类型可以加特殊符号'-'，比正常电话11位号码多一位
    var maxLength = /\-/g.test(this.value) ? 13 : 11;
    $(this).val((this.value.split('-').length > 2 || this.value.
length > maxLength) ? this.value.substring(0, this.value.length
- 1) : this.value.replace([3, 4].indexOf(this.value.indexOf('-')
) < 0 ? /[^\d]+/g : /[^\d-]+/g, ''));
}).on('input', 'input[data-type=money]', function () {
    //只能输入浮点数字类型
    var _val = this.value.replace(/[^.\d]+/g, '');
    if (/(^\.+)|(([^]*[.]){2})/.test(this.value)) {
        var _regex = /\d+(\.\d*)?/.exec(this.value);
        _val = (_regex && _regex.length) ? _regex[0] : this.valu
e.substring(0, this.value.length - 1);
    }
    $(this).val(_val);
});

// 去掉input框的空格
$('input[type=text]').on('blur', function () {
    $(this).val($.trim($(this).val()));
});
```

kungeekUI插件触发

```
$(document).ready(function () {
    // 触发数据表格
    $('table.kungeekui-datagrid').each(function () {
        new Datagrid({
        target: $(this)
        });
    })
    // 触发日期选择框
    if (window.moment) {
        $('input.kg-dateSelect[type="text"]').dateSelect();
    }
})
```

bootstrap组件触发

在这里默认触发bootstrap的组件功能。

```
//页签
$('#navTabs a').click(function (e) {
    e.preventDefault()
    $(this).tab('show')
})
//显示更多
$('.crm-show-more').on('click', function () {
    $(this).toggleClass('expand');
})
```